XINS: The Anatomy of an Indoor Positioning and Navigation Architecture

Yuan Gao[*]
gaoy03@gmail.com

Qingxuan Yang[†]
qingxuan@google.com

Guanfeng Li[†]
ligf@google.com

Edward Y. Chang[†]
edchang@google.com

Dong Wang[†]
wangdong@google.com

Chengu Wang[‡]
wangchengu@gmail.com

Hang Qu[†]
quhang@google.com

Pei Dong[†]
dongpei@google.com

Faen Zhang[†]
zhangfaen@google.com

[†]Google Research
Beijing, China

[*]Dept. of Computer Science and Technology
Tsinghua University
Beijing, China

[‡]IIIS
Tsinghua University
Beijing, China

ABSTRACT

Location-Based Service (LBS) is becoming a ubiquitous technology for mobile devices. In this work, we propose a signal-fusion architecture called XINS to perform effective indoor positioning and navigation. XINS uses signals from inertial navigation units as well as WiFi and floor-map constraints to detect turns, estimate travel distances, and predict locations. XINS employs non-intrusive calibration procedures to significantly reduce errors, and fuses signals synergistically to improve computational efficiency, enhance location-prediction accuracy, and conserve power.

ACM Classification Keywords

C.3 Special-purpose and application-based systems: Real-time and embedded systems.

General Terms

Algorithms, Design, Experimentation, Measurement, Performance.

Author Keywords

calibration, inertial tracking, particle filters, location

INTRODUCTION

Resent years have seen the number of "smart" wireless devices such as mobile phones and iPad-like computers grow rapidly. Being able to keep track of locations of moving devices can enhance a number of applications. Therefore, Location-Based Service (LBS) is quickly becoming the next ubiquitous technology for a wide range of mobile applications, such as location positioning, location navigation, location-aware search, commerce, and advertisements, just to name a few. Current LBS technologies, however, suffer from at least two main shortcomings. The first is the lack of support for indoor positioning and navigation. The second is that the power consumption of receiving GPS and WiFi signals is too high to continuous use.

In this work, we propose a signal-fusion architecture to address the above shortcomings. We name our architecture XINS, where X stands for a signal source that can calibrate a moving object's location as an external reference. An X can be GPS, WiFi, or location information of a nearby device transmitted over a P2P protocol. The INS in XINS stands for a *inertial navigation system*, which includes, but is not limited to, motion-sensing devices such as accelerometers, gyroscopes, and magnetic sensors. Modern phones like the iPhone and Google Nexus S are equipped with tiny and energy-efficient INS'. The design goals of XINS are (1) to fuse signals from Xs and INS' to perform accurate location positioning, both indoor and outdoor, and (2) to do so in a power-conserving way.

System architectures similar to XINS have been developed before, but with vastly different constraints and at much higher costs. For instance, aircrafts, guided missiles, and submarines have been designed to compute positions and velocities using external references and inertial navigation systems. A mobile phone, though, while equipped with similar motion-sensing devices, can not compute positions and velocities as accurately as the former due to several factors. These include low manufacturing quality originating from cost constraints, environmental noise, and the dynamic motions of people carrying the devices (as opposed to an aircraft for example). Moreover, the power usage of acquiring X signals on a mobile phone is much higher as a percentage of total energy consumed compared to that used on e.g., a submarine. These factors make the design of XINS exceedingly challenging.

XINS consists of three major components. In addition to X and INS, XINS keeps a map of the area it tracks. For indoor scenarios, XINS keeps a 2.5D map depicting the floor plan

MLBS'11, September 18, 2011, Beijing, China.

of a building. A 2.5D map suffices because XINS tracks only the floor level, not the height of the device vertical to the floor. XINS divides each floor into multiple *areas*, and identifies *landmarks* between them. Our design goal is to track a mobile device at a landmark and in an area with high accuracy. The key ideas of XINS are as follows:

1. *Detect turns, floor-changes and estimate travel-distances*. XINS collects signals from accelerometers, gyroscopes and compasses. Using these signals together with a floor map, XINS determines a user's travel distance and if they have turned or made a floor change.
2. *Detect positions*. XINS conducts joint inference by considering signals of X and INS, together with an indoor map, to determine a mobile user's location.

To implement these ideas, XINS faces two daunting technical challenges. First, inexpensive inertial navigation systems are notorious for producing errors due to misalignment, zero bias, and integration drift. A slightly inaccurate reading on linear acceleration (accelerometers) and angular velocity (gyroscopes) can be integrated into progressively (in time) large errors in velocity, which are compounded into greater errors in position. Worse yet, the motion of a pedestrian is "non-smooth" compared to an aircraft. For instance, a mobile device can be carried by different people at different time and in different manners. It is virtually impossible to tell true signals from noise. To eliminate noise, we devise non-intrusive calibration schemes. For accelerometers, we propose using an optimization framework (combining the Nelder-Mead method and gradient descent) to transforme an ill-formed ellipse (due to noisy readings) into a sphere with a radius based on the acceleration of gravity (i.e., $9.8015m/s^2$). The information obtained from the ellipse-to-sphere transformation is then used as parameters to calibrate future readings. For gyroscopes, we propose the Vibration Energy Model to determine a pedestrian's moving direction based on the *Equipartition theorem*. These calibration processes lay down a solid foundation for XINS to obtain accurate sensor readings so as to conduct productive multimodal signal fusion to perform location positioning. In this work, we use *particle filters* as our fusion algorithm. We show how we can intelligently use constraints from floor maps and landmark hints to drastically reduce the number of particles needed to perform accurate position prediction.

In summary, the contributions of this paper are as follows:

1. We propose XINS, which uses particle filters to fuse signals from different sources. We show that the usage of some signals as constraints can make particle filters work much more efficiently.
2. We devise non-intrusive INS calibration schemes to reduce sensor-reading errors.
3. We propose Vibration Energy Model (VEM), which reduces the number of integrals from three to one, by computing in the dual, energy domain (thanks to the Equipartition theorem); and therefore, avoids small errors from being magnified.
4. We show that XINS can achieve location prediction in a power conserved way because sample rates on power-consuming WiFi and GPS can be reduced by employing INS to fill the gaps.

The structure of this paper is as follows. We first review related work. Then we present our proposed XINS. Following that, we show experimental results of XINS. Finally, we offer our concluding remarks.

RELATED WORK

LBS was first deployed in the turn of the century by Palm VII, Swisscom, Vodafone, and DoCoMo. These first wave of deployment performed location positioning based on the locations of the nearby cell towers. The accuracy of such an approach ranges from one hundred to a few thousand meters, depending on the density of cell towers. In 2004, the Global Positioning System (GPS) was tested successfully to work with a mobile phone by Quadcomm. GPS is now available on most smart phones and can achieve outdoor location positioning with approximately ten-meter accuracy. Its major shortcomings are high power consumption, long TTFF (time to the first fix), and unavailability in urban tunnels and indoor environments.

We survey related work in two areas: *indoor positioning* and *INS calibration*.

Indoor Positioning

WIFI-Based Schemes

WiFi has been shown to achieve three- to ten-meter indoor positioning accuracy [9, 20]. The achievable accuracy depends on two factors: access point density and the location-positioning algorithms employed. Location positioning algorithms can be divided into two approaches: signal propagation and signal heat maps. The propagation approach infers the distance of an object to an access point based on signal strength. A widely used equation to estimate distance based on signal strength is the Friis transmission equation. However, due to environmental factors such as antenna obstruction (a phone can be placed in a purse or in a pocket) and antenna misalignment, the Friis equation may not yield an accurate estimate. Beyond this, the power at both an antenna and a mobile device may be unknown.

The signal heat map approach [6, 9] constructs Radio Frequency (RF) signatures at indoor locations. Based on an RF signature, a mobile device can predict its possible locations. To pin down a particular location with the highest probability, the work of [12] proposes using motion history information based on the fact that a person cannot travel a long distance in a short period of time. The work of [3] introduces geometric constraints. And [8] uses expensive image processing techniques to infer locations from photos taken. The major shortcoming of the signal heat map approach is the laborious site survey cost and time. In addition, when the WiFi configuration changes, the affected area must be resurveyed.

INS-Based Schemes

As mentioned in the Introduction Section, inertial navigation systems have been widely used in e.g., aircraft and guided missiles. The INS' installed on those apparatuses can afford high costs and hence achieve high precision. Here, we focus our survey on using inexpensive INS units on indoor positioning.

Using INS units to localize a person is not new. Related work can be broadly classified into two categories according to how INS units are placed. The work of Woodman and Harle [18] implements an indoor pedestrian-localization prototype using a foot-mounted inertial unit. The work of [11] assumes an INS unit to be wore on the waist. Both schemes make sure that the motion patterns of their INS units are predictable. In the foot-mounting case, when a foot touches the floor, the INS unit receives a clean signal to start a stride, and this reset is done periodically. In the waist-wearing case, the INS unit is expected to move steadily. However, INS units embedded in a mobile phone cannot enjoy these advantages. A mobile phone can be worn on the waist, placed in a purse, put in a pocket, or swang in a hand.

The work of [4] develops an on-phone pedometer using an accelerometer. Though the estimated steps may not be very reliable, it provides a good estimate on travel distance (but without orientation). A compass can be used to estimate the user's orientation, but a reliable reading requires the user to hold the phone in such a way that the orientation of the phone always agrees with that of the user. Also, compasses on phone are notoriously inaccurate due to severe inference from the phones' other electronic units. Pack [16] adopts the theory that double integration of acceleration gives the displacement, but it only works when the orientation of the phone does not change too frequently. Pack also utilizes the GPS signal to periodically correct the estimated velocity and position.

All these methods which make use of INS do not yet have an effective strategy to combat the progressive errors introduced by various INS noises.

INS Calibration

High-end sensor calibration usually requires an expensive mechanical platform, by which orientation and rotation speed can be precisely controlled. The sensor outputs are then compared with external calibration values calculated by known parameters of the mechanical platform. This kind of high-cost and intrusive approach is not suitable for calibrating INS units on mobile phones.

To address mobile-phone INS calibration, inexpensive equipment such as optical trackers [10] were proposed and experimented with. However, it is not realistic to ask a phone buyer to also purchase such a calibration station, or travel to a shop to periodically calibrate their devices. The work of [19] employs a gravity-based approach, which requires collecting a number of measurements to calibrate accelerometers. Olivares [15] suggests that the information learned from the

Figure 1. The 2.5D map of two floors in our site.

the calibration of accelerometers can help calibrate gyroscopes. In this work, we ensure that calibration can be performed effectively with minimal effort required of mobile-phone users.

XINS

This section depicts the four key components of XINS: *floor map*, *X signals*, *INS signals*, and *signal fusion*. The fusion component considers signals from the other three components to perform location prediction.

Indoor Map

Figure 1 shows a two-story floor plan used in our experiments, in which travel routes, rooms, entrances, landmarks, stairs, and elevators are marked. Furthermore, we model elements on the indoor map by several data structures, including levels, sections, polygons, points, segments, intersection paths and angles.

Beyond the benefit of rendering the indoor graphic map to present to the user, the key mission of the indoor map component is to provide constraints to our fusion module described later in this paper. The constraints are provided in two key APIs. API $\mathrm{GetCrossPossibility}(A, B)$ returns the possibility of moving from point A to point B in a short period of time according to the map. For example, the possibility of going from A, on one side of a solid wall, to B, on the other side, is $zero$; if A and B are separated by a small table, the possibility may be, say, 0.5, while it is entirely possible to transit from A to B in a large open space. The *Map component* also incorporates information from *WiFi component* and *INS component* to identify landmarks to further reduce the number of particles in our filter. Landmarks are polygons lying on a special part of the map, such as the junction of two aisles. A user located within these special polygon areas can be identified with high confidence due to the unique properties of the polygon (e.g., its shape) combined with information obtained from the WiFi and INS components. API $\mathrm{GetLandmark}(P, WiFi, Map)$ outlined in Algorithm 1 is used to identify a landmark.

Algorithm 1 Get the landmark that covers a point P

```
0: Procedure GetLandmark(P, WiFi, Map)
1: // Get a set of pairs ⟨pr, possibility⟩, which is a location
2: // and its probability with that WiFi signature.
3: S ← setof⟨pr, possibility⟩
4: result ← null
5: for each item in S do
6:     if pr ≥ threshold then
7:         if P is at some special place on a map && INS reading doesn't
           contradicts the fact then
8:             result ← the polygon at P
9:             break;
10:        end if
11:    end if
12: end for
13: return result
```

X Signals and Landmark Detection

X signals can be GPS and WiFi. GPS signals are available at the entrance into a building or in areas near windows. X signals play an important role in reducing the number of particles in our filter. Let us specifically focus our discussion on WiFi signals only. As discussed in the related work section, WiFi-based location predication has two basic methods, wave propagation, and signal heat map. However, in indoor environments, the former generally does not work because the WiFi signal propagation is usually heavily dependent on the surrounding complex environment. WiFi signals are seriously interfered by solid walls, cubic walls and furniture. This situation becomes even worse if we consider dynamic effects such as people walking, etc. We employ a slightly revised signal heat map approach. The key difference is that, in addition to the standard signal heat map approach, we identify landmarks, around which key navigation decisions such as stops and turns are made. When conducting access-point configuration and site survey, we can pay special attention onto these selected landmarks to improve their detection accuracy. Furthermore, around a landmark, XINS can conduct joint inference with X and INS signals. For instance, a turn can be determined to be 90° to the left even though INS reports it to be a 80° because the landmark provides such a navigation constraint. Details are discussed in the fusion section.

The X component provides two APIs.
$\mathrm{GetWiFiTransitPossibility}(A, B)$ returns the possibility of transit from A to B according to the strength of WiFi hot spot signals. This algorithm is outlined in Algorithm 2.

In Algorithm 2, $\mathrm{GetWiFiProbabilityAt}(P, WiFi_P)$ is a key procedure that *WiFi component* defines, which returns the probability of a WiFi signature $WiFi_P$ at point P. Notice that we give particles located at or near landmarks preferential treatment when choosing the position of a queried mobile device.

Inertial Navigation Systems

As the MEMS inertial sensors embedded inside mobile phones have a large noise floor, they must be calibrated before being used for localization. There are two opportunities in which the inertial units in a mobile phone may be calibrated: at the manufacturer and at home. The calibration process at the manufacturer can rely on external, expensive devices. However, once a user has purchased a phone, the calibration process at home should be one time, non-intrusive, and certainly cannot rely on external devices such as a turn table.

Algorithm 2 Compute the probability of transit from area covering A to covering B

```
0: Procedure GetWiFiTransitPossibility(A, B)
1: // get the corresponding landmarks.
2: L_A ← GetLandmark(A, WiFi_A, Map)
3: L_B ← GetLandmark(B, Wifi_B, Map)
4: p_A, p_B ← 100%
5: if L_A == null then
6:     p_A ← GetWiFiProbabilityAt(A, WiFi_A)
7: end if
8: if L_B == null then
9:     p_B ← GetWiFiProbabilityAt(B, WiFi_B)
10: end if
11: return p_A · p_B
```

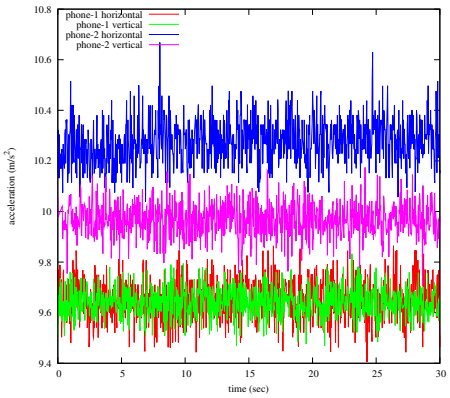


Figure 2. The magnitudes of the acceleration measured by two phones in two placements. The top two rows (blue and pink) are signals of Phone 1 and the bottom two rows (red and green) are those of Phone 2.

Accelerometer Calibration

We first discuss the error model for calibrating accelerometers, and then detail the calibration process.

The accelerometer embedded in the Nexus S suffers from a large noise floor. A simple experiment helped us analyze and partition the noise floor with the aim of modeling it. Two Nexus-S phones were placed side-by-side, first lying down on a desk, and then standing against a wall. The most sensitive axis of the 3-axis accelerometer is that which is closest to being parallel with the pull of gravity. The aforementioned two placements allow us to isolate which axis we wish to be the most sensitive, but do not require the phones or any axis of the accelerometers to be precisely parallel or perpendicular to the horizontal. In each placement, the phone remains stationary for at least 30 seconds. The magnitude of the acceleration measured during these 30 seconds are shown in Figure 2. We observe three significant types of error as follows:

- ***Random error.*** Although both phones were stationary, the magnitudes of the acceleration measurements still jittered quite noticeably, which illustrates the existence of random noise.

- ***Bias error and scale factor error.*** When the two phones were placed at the two different positions, Figure 2 indicates that both suffered from bias and scale factor error, because the magnitudes of the accelerations are different both between the phones and between their positions, namely, *i)* the magnitude of the acceleration measured by the second phone is at least $0.2m/s^2$ larger than that of the first phone; *ii)* the same phone measured different magnitudes of accelerations when the most sensitive axis with respect to the gravity was changed; *iii)* the magnitudes of the accelerations measured on the two Nexus' both deviated from the acceleration of gravity (i.e., $9.8m/s^2$). Therefore, each phone must be calibrated individually.

We model these three accelerometer errors in Eq.1, in which the 3-axis accelerometer reading a_{raw} is calibrated to a_c, where a_{raw}, a_c, b and V are 3 dimensional vectors, and S is a 3×3 diagonal matrix $\mathrm{diag}[S_x, S_y, S_z]$.

$$a_c = S(a_{raw} - b - V), \tag{1}$$

This model aims at getting rid of the random error V, bias b and scale factor error S in a_c.

The most important property used for calibration is that the magnitude of the acceleration measured while the phones are stationary must equal that of gravity [7]. Thus, if no errors, the measured acceleration is on a sphere whose radius is the magnitude of the acceleration of gravity. The biases and scale factors stretch the sphere into an ill-formed ellipse, and the random errors perturb the ellipse. The aim of our calibration process is to determine the parameters in the error model to transform the ellipse back into the desired sphere.

Since the random error V is not a constant value and it differs from time to time, an efficient method to eliminate the randomness is averaging the accelerometer readings during a time window. Allan Variance [1] is adopted to determine how long the time window should be. Allan Variance was first proposed by David Allan to measure oscillator instability in the time domain, and it is also an efficient method for representing random noise as a function of average time [7]. Here, the *average time* means the span of time during which samples are averaged together. The definition of Allan Variance is

$$\sigma_y^2(T) = \frac{1}{2} \sum_{i=1}^{N} [y(i,T) - y(i-1,T)], \tag{2}$$

where $y(i,T)$ is the i^{th} average, and each average spans T seconds. We collected stationary accelerometer signals for 160 minutes. The average time T varies from 1 second ($N = 9599$) to 400 seconds ($N = 23$). Figure 3 only shows the Allan Variance when the averaging time varies from one to 15 seconds, and that of $T > 15$ sec doesn't change significantly. The Allan Variance of z-axis of the accelerometer embedded in the test mobile phone takes the longest time

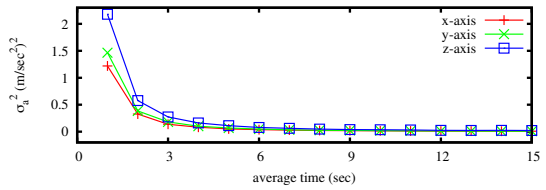


Figure 3. The Allan variance of the 3-axis accelerometer in the Nexus S.

interval of 6 seconds to converge. This implies that the calibration of the Nexus-S requires the phone to be stationary for at least 6 seconds to obtain a stable average value to eliminate the random noise so that the average values will not change significantly in the any of the following 6 seconds intervals. The average time longer than 6 seconds will surely reduce the Allan Variance but in an insignificant way, however, the in-field calibration requires the calibration time as short as possible, and thus 6 seconds is chosen as an appropriate value. This random noise removal step differentiates our calibration process from that proposed by [7], which arbitrarily chooses a one second interval as the length of averaging time window. The measured Allan Variance indicates that the average of accelerometer signals spanning only one second differs from each second to the next.

After random noise has been removed by averaging acceleration readings spanning at least 6 seconds, the next step of calibration is to determine biases and scale factors, i.e., the six parameters $(S_x, S_y, S_z, b_x, b_y, b_z)$.

As there are six parameters, it is prudent to have enough data to make an over-determined system. Data can be collected in users' daily lives to calibrate the accelerometers in a non-intrusive way. We use the same quasi-static detector s that used in [7]. When we detect a static state lasting for at least 6 seconds, the average of 3-axis accelerometer signals forms *a calibration data set*. In our daily lives, there are myriad chances that a phone stays stationary, e.g., on the desk, in users' pockets when sitting or standing still, etc. Given n sets, the calibration is to compute $(S_x, S_y, S_z, b_x, b_y, b_z)$ by using a cost function similar to [7]:

$$f(S,b) = \sum_{i=1}^{n} (\|a_{ci}\| - \|a_g\|)^2 \tag{3}$$

The cost function $f(S,b)$ is minimized using a combination of Nelder-Mead method [14] and gradient descent [17]. The reason we added gradient descent is because Nelder-Mead method sometimes missed the global minimum when initialized with different values. We used the result of the Nelder-Mead method as the initial start value to the gradient descent method, and used the result of the gradient decent to seed the next round of Nelder-Mead. After several iterations, the minimization procedure is completed.

Although calibration data sets with gravity vector measurements spread evenly over the sphere will perform a better calibration, the non-intrusive calibration aims at enabling

calibration at home, which cannot require too many constraints on calibration data sets. In the Experiments Section, we collect 50 sets of data, out of which 12 sets are chosen at random. We validate that 12 sets are enough to effectively train the calibration scheme to obtain the six required calibration parameters.

Tracking mobile devices users with INS

After the accelerometers have been calibrated, we can begin to track a user with a mobile device equipped with INS units. We track the direction as well as the distance of the pedestrian's movement.

Direction is derived through the Vibration Energy Model, and the distance is computed by multiplying the number of steps by the length of the step, both of which will be described below.

Vibration Energy Model

The Vibration Energy Model (VEM) aims at calculating the direction of the users' movement from the INS units in swinging hands instead of having to be mounted on any part of the users. VEM is based on *Equipartition theorem*, which states that energy is shared equally among all degrees of freedom. When humans walk, they usually swing their arms parallel to the direction they are walking. If a pedestrian holds a phone without swinging his/her arms, he/she will swing his/her body parallel to the direction he/she is walking in order to keep the balance. In an extreme case, even if a pedestrian wants to walk crosswise, like a crab, he/she will still tend to swing his/her body like the crab, i.e., also parallel to the walking direction. This phenomenon can be explained theoretically a pedestrian stores energy for walking while swinging her arms. The swing is regarded as an energy pool. Following the *Equipartition theorem*, pedestrians can easily use/store energy between walking/swinging. VEM uses the signal of swinging energy as a hint to predict the direction.

VEM begins with the accelerations in the *absolute reference frame* (denoted by the vector $A_a = (A_{ax}, A_{ay}, A_{az})^T$) to derive the direction of the pedestrian's movement. In the absolute reference frame we used in this work, the x axis points to the east, the y axis points to the north and the z axis points towards the ground. However, the accelerometer inside the mobile phone provides the acceleration in the mobile phone reference system. The magnetic field measured by magnetic field sensor and the gravity measured by the accelerometer are both used to determine the initial orientation of the mobile phone. During the tracking, gyroscopic measurements are adopted to update the orientation.

The direction of the pedestrian's motion is determined by VEM as follows:

1. $A_{high} = \mathrm{HPF}(A_a)$. A_a is fed to a high pass filter to get the high frequency component, A_{high}, which retains the swing energy. The information along the z-axis of the absolute reference frame is dropped because we aim at determining the direction of the pedestrian on the horizontal plane.

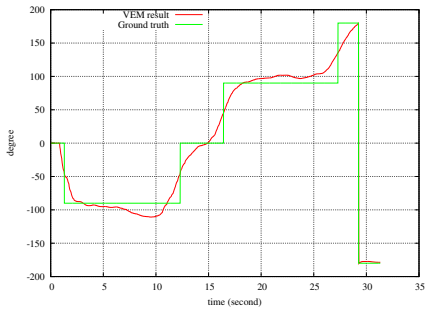


Figure 4. Using VEM to detect the direction of the users' movement (Considering the map constraint). The first one second shows 0° because of determining the initial rotation matrix.

2. $A_{energy} = A_{high} \times A_{high}^T$. A_{energy} is a 2×2 symmetric matrix, which can be represented as

$$A_{energy} = \begin{pmatrix} a^2 & ab \\ ab & b^2 \end{pmatrix}.$$

   Here, energy refers to the stored elastic potential energy, which is proportional to acceleration.

3. $A_{lowenergy} = \mathrm{LPF}(A_{energy})$. A simple linear low pass filter is used to extract the low frequency component of A_{energy}, i.e., to smooth the A_{energy}. The $A_{lowenergy}$ can be represented as following because the filter is linear:

$$A_{lowenergy} = \begin{pmatrix} \mathrm{LPF}(a^2) & \mathrm{LPF}(ab) \\ \mathrm{LPF}(ab) & \mathrm{LPF}(b^2) \end{pmatrix}.$$

4. The projection of $A_{lowenergy}$ on the movement direction is the largest among all the directions, i.e., the α which maximizes $f(\alpha) = \mathrm{LPF}((a\cos\alpha + b\sin\alpha)^2)$ is the angle of the pedestrian's direction of movement. Using $\frac{df(\alpha)}{d\alpha} = 0$, we can easily derive that

$$\alpha = \frac{1}{2}\arctan\frac{2\mathrm{LPF}(ab)}{\mathrm{LPF}(a^2) - \mathrm{LPF}(b^2)} + k\cdot 90^\circ, k = 0, 1, 2, 3.$$

   Note that two of these α actually minimize $f(\alpha)$, which can be checked by substituting these values of α back into $f(\alpha)$. The remaining two values of α are both the possible directions of motion as VEM is only able to detect the direction of swinging. This swinging direction is parallel to the direction of walking, but may be opposite to it. The *signal fusion* component involves the constraints from both WiFi and the floor map to decide the direction of motion.

A mobile phone user, holding the phone and swinging arms while walking, walks around a block, first heading south, and then east, north and west. Figure 4 shows the direction of the user detected by VEM (the red line), which closely matches the ground truth (the green line).

Step detection

While pedestrians walk on level ground, measured accelerations show evident rhythms as can be seen in Figure 5. Further than [4], we evaluate that rhythms exist when mobile

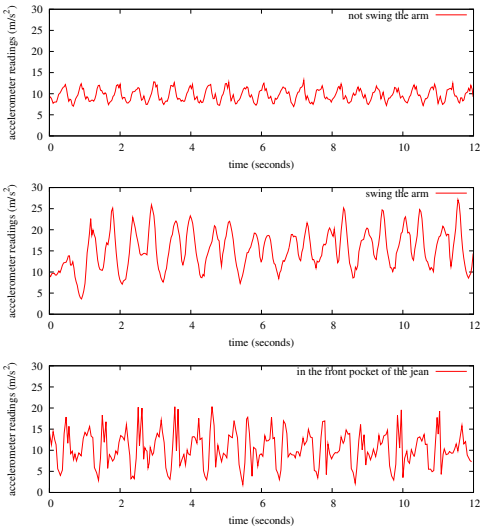


Figure 5. The measured accelerations of the pedestrian, with the phone in a steady hand (top), with a swinging arm (middle) and in the front pocket of user's jeans (bottom).

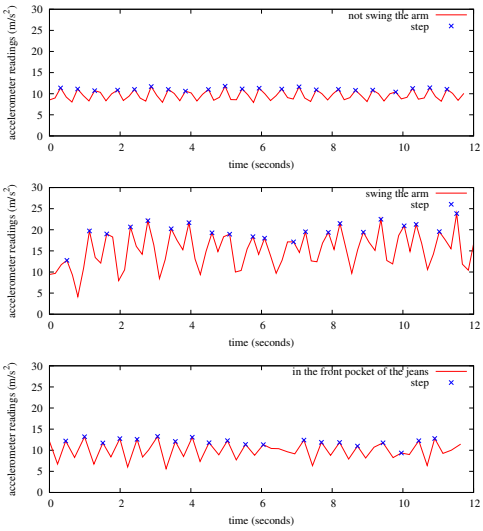


Figure 6. The step detection of the pedestrian, with the phone in a steady hand (top), with a swinging arm (middle) and in the front pocket of user's jeans (bottom).

phone equipped with accelerometer is in different positions in Figure 5, i.e., held in a pedestrian's hand when they swing as they walk (top), held in a pedestrians's hand while they hold the phone steady in front of their positions (middle), and put in the front pocket of the pedestrian's jeans (bottom). Only the magnitudes of the accelerations, which have been calibrated with the parameters got in the calibration section, are shown in the figure. The sample rate of the accelerometer is about $30Hz$.

As the walking of pedestrians is a cyclic movement composed of leg-striking, and toe-off [13], each placement of the accelerometer shows a rhythm. However, the rhythms have different presentations in different placements. When the pedestrian holds the phone in the hand as her arms swing for balance, the acceleration measurements show one spike corresponding to each step, while there are two close spikes corresponding to each step in the other three scenarios. The difference in the number of spikes is because most people firstly settle the toe back on the ground and then the heel, which will generate two spikes, but if the phone is held in the hand of a swinging arm, the impact of the toe-heel back to the ground is not evident. We design a smooth-spike algorithm to detect each step no matter how the pedestrian places the phone. Each of the N samples of the raw accelerometer readings are firstly calibrated using Equation 1. Besides eliminating the biases and scale factors error as well as reducing random error, the calibration also integrates two close spikes into just one spike, and thus the calibration is called *smooth*. The smoothed accelerations then form a sequence (denoted as a_{si}) and are sent to the spike detect algorithm. If a_{si} is larger than a_{si-1} and a_{si+1}, a_{si} is regarded as a spike, and marked as a single step. In practise, $N = 5$ seems to be the best option, as nearly no useful spikes are smoothed, while the two spikes caused by toe-heel movement are integrated as one. Although the smooth-spike method introduces lag by averaging only after N-samples have been taken, $N = 5$ and $30Hz$ sample rate makes this lag around $\frac{1}{6}$ second, which is acceptable. Figure 6 shows the step detection results of the three different phone placements. In the above experiment, only one step is missed when the phone is in the front pocket of the jeans. The step size is approximately as the work of Constandache etc. [4].

Fusion Using Particle Filters

We separate the detection of floor-change from predicting positions on a fixed floor. For a floor-change detection, we use VEM to track a user's movement between floors. If VEM detects consistent vertical energy, we can assume that the user is moving on a staircase, escalator or elevator. Once the user has reached the desired floor (vertical energy has subsidized for a empirical chosen time T), we consider the floor change complete. A landmark on that floor can provide confirmation.

To track locations on a floor, we employ Particle Filter as our fusion algorithm. Particle filters are usually used to estimate Bayesian models in which latent variables are connected in a Markov chain, where the state space of the latent variables is continuous (rather than discrete), and not sufficiently restricted to make exact inferences tractable. Under the Markov assumption, later events are influenced by prior ones, allowing a Bayesian filter to track the state of a dynamic system through time. A good tutorial regarding particle filters can be found in [2]. However, particle filters are known to be computationally intensive as the state space of latent variables grows, resulting in a huge number of particles. To make the framework computational feasible on mobile devices such as cell phones, we use floor plans and WiFi landmarks as constrains to restrict the number of particles.

In our localization framework, at time t a particle has a state $s_t = (x_t, y_t, \theta_t, landmark_t)$, where (x_t, y_t) and θ_t are the horizontal position and the heading direction of the user, respectively. $landmark_t$ is a special polygon area where we can get strong indicators from the map, WiFi and INS units. This polygon is then considered the presumed location of the user. Note that $landmark_t$ can be $null$ if the user is not near any of the landmarks. Since we detect which floor the user is on separately using VEM, it is not necessary to store the altitude in this state variable. The input to our particle filter includes the floor plan, the readings from INS', the WiFi signal readings, and the WiFi signal heat-map from a site survey. The output is a winning particle, s_t, i.e., a user indoor location.

We now outline the propagation, correction and resampling steps of the particle filters in our localization framework in detail.

Propagation

The state propagation step is similar to that of [18], and we briefly outline it here:

$$\theta_t = \theta_{t-1} + \delta\theta' \quad (4)$$

$$x_t = x_{t-1} + l' \cdot \cos\theta_t \quad (5)$$

$$y_t = y_{t-1} + l' \cdot \sin\theta_t \quad (6)$$

where l' and θ' are the step length and heading direction, respectively, measured by the INS units with Gaussian disturbance caused by inaccuracies by the INS units taken into account. Let us define a *disk* as the area covered by the signal of a WiFi access point. To update $landmark_t$ in the state variable, we use step vector AB, where $A = (x_{t-1}, y_{t-1})$ and $B = (x_t, y_t)$ and consider two scenarios:

1. Vector AB intersects with no disks of known landmarks. The particle must remain in the vicinity of the same landmark as it was at previously (this includes the case when neither A nor B is covered by any landmarks, i.e., $landmark_t$ is $null$).
2. Vector AB intersects with one or more landmark disks. The $landmark_t$ is changed accordingly.

We unify the above two scenarios in the API GetLandmark() in the *Map component*.

Correction

The correction step assigns a weight w_t to a propagated particle. Here we employ the APIs from *Map component* and *WiFi component* as constraints to reduce the number of particles. Let $p_{wall} = \text{GetCrossPossibility}(A, B)$ and $p_{wifi} = \text{GetWiFiTransitPossibility}(A, B)$. We assign the particle a weight $w_t = p_{wall} \cdot p_{wifi}$. The combination of the propagation and correction steps generates a particle at time t from a sampled state at time $t-1$. This process is summarized in Algorithm 3.

Resampling

Similar to what was proposed in [18], we use KLD-sampling for the resampling step. The basic idea of KLD-sampling is to generate a number of particles at each step such that the approximation error introduced by using a sample-based representation remains below a specified threshold.

Algorithm 3 Update a particle from time $t-1$ to time t.

0: **Procedure** Update(s_{t-1}, u_t)
1: *// get the new position.*
2: $\theta_t \leftarrow \theta_{t-1} + \delta\theta'$
3: $x_t = x_{t-1} + l' \cdot \cos\theta_t$
4: $y_t = y_{t-1} + l' \cdot \sin\theta_t$
5: *// initialize the intersection algorithm.*
6: $A \leftarrow (x_{t-1}, y_{t-1})$
7: $B \leftarrow (x_t, y_t)$
8: $p_{wall} \leftarrow \text{GetCrossPossibility}(A, B)$
9: $p_{wifi} \leftarrow \text{GetWiFiTransitPossibility}(A, B)$
10: *// get the weight of particle at time t.*
11: $w_t \leftarrow p_{wall} \cdot p_{wifi}$
12: *// get the new landmark.*
13: $landmark_t \leftarrow \text{GetLandmark}(B, WiFi_B, Map)$
14: $s_t \leftarrow (x_t, y_t, \theta_t, landmark_t)$
15: *// return this particle.*
16: **return** (s_t, w_t)

EXPERIMENTS

Evaluation of the accelerometer calibration

In this section, we want to answer two questions:
1. How many data sets should be used for accelerometer calibration?
2. Does calibration effectively correct the errors of accelerometer measurements?

For ideal accelerometer calibration that transforms an ill-formed ellipse into a sphere (with the radius corresponding to the acceleration of gravity), the variance of the N samples distributed in the calibrated sphere should be $zero$. Therefore, we can evaluate the calibration by means of measuring the variance of the magnitudes of the calibrated accelerations in the test sets. The smaller the variance is, the better the calibration is.

To answer the first question, we collected a total of 50 sets of data in the same way as collecting the *calibration data set* described in the calibration section. Each time, n sets were randomly picked as the calibration data sets to determine the biases and scale factors using the Nelder-Mead method and gradient descent method, and the remaining $(50 - n)$ sets were used as the test sets. For each n, the experiment was repeated 5 times. Figure 7 shows the average variances of the calibrated accelerations when n was set different, as well as the variance of the variances. Generally speaking, the variances of the calibrated accelerations decrease as n increases. However, calibration data sets containing more than 12 data sets do not significantly improve calibration performance, and thus we adopt 12 as the size of calibration data sets.

To investigate the second question, we compare the variance of the test data sets before and after the calibration using the 12 sets of data. The phone used for experiment is the same as the Phone 2 in Figure 2. The variance of the remaining 38 sets of test data drops from more than 0.17 to around 0 (the average of the variance after calibration is 1.06×10^{-3}).

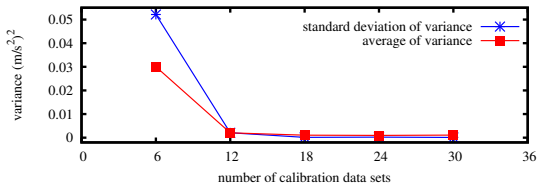


Figure 7. The calibration performance using different sizes of calibration data sets. The red line shows the average of the variance, and the blue line shows the standard deviation of the variance.

Evaluation of Localization using XINS

Locating mobile devices with XINS aims at generating the current position of the users; we evaluate both the correctness of INS tracking and the effectiveness of particle filters.

We define a set of experiments with different routines and different placements of the mobile phone:

- ***The routines.*** Totally two sets of different routines are used in the experiments, namely, a $5.35m \times 13.35m$ rectangle around which the pedestrian is to walk in a clockwise and counterclockwise direction (denoted as *clockwise-rect* and *counterclockwise-rect* respectively). No matter in a clockwise or anticlockwise way, the user started from the southwest corner of the rectangle.
- ***The placements.*** The mobile phone is placed in three different configurations, namely, a static hand, a swinging hand and the front pocket of the user's jeans. Here, a static hand is the scenario of a pedestrian trying to watch the screen of the phone while walking.

There were totally 6 sets of experimental data (two different routines with three different placements of the phone). We plotted a trace with the positions generated by our XINS at the rate of $1Hz$, and examined how well the trace matches the actual routines. The southwest corner is assumed as $(0, 0)$ in the figures.

We show the results of the experiments in Figure 8. The results illustrate the effectiveness of the INS component as well as the particle filters. It can be seen that VEM correctly detects turns. Figure 8(e) and 8(f) show that, in some cases, while the clockwise/counterclockwise movement is correctly detected by only employing the INS units with VEM, the trace of motion is displaced from the actual routine. This is because VEM can only detect the direction parallel to that of the pedestrian, but without knowledge about whether walking forward or backward, randomly choosing one direction from the possible two. However, the particle filters successfully correct this uncertainty introduced by VEM. Generally speaking, particle filters are able to provide accurate localization by fusing the signals from floor maps, WiFi and INS. The red lines match well with the $5.35m \times 13.35m$ rectangle in all three placements of the mobile phones as well as the two different routines.

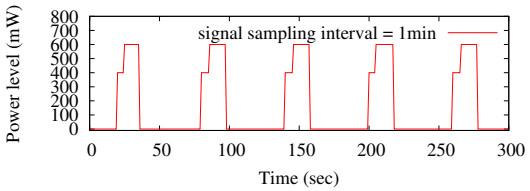


Figure 9. The power consumption pattern of X component.

Discussion on the energy consumption of XINS

In the indoor environment, the X signal mainly relies on WiFi, and the INS fill the gap between any two successive X signals, which makes it possible to increase the interval of WiFi. Note that, although we use a GPS power consumption to illustrate, other X signals like WiFi exhibit a similar power usage pattern.

Zhuang, etc. measured the instantaneous power-spikes of GPS sensing of the Android phone [21]. They stated the GPS invocation is composed by a locking period (which spans $4-5$ seconds and averagely consumes about $400mW$ energy) and a sensing/reporting period (which spans $10-12$ seconds and averagely consumes about $600mW$). Based on these data, Figure 9 illustrates how the power spikes distribute. With the interval of GPS activated increases, the energy consumption decreases. The work of [5] investigated the energy consumption of GPS and WiFi separately using a Nokia N95 phone, which shows a similar result like the Android phone. Consequentially, the increase of the interval, at which the WiFi is invoked for requiring location in the Android phone, also decreases the energy consumption.

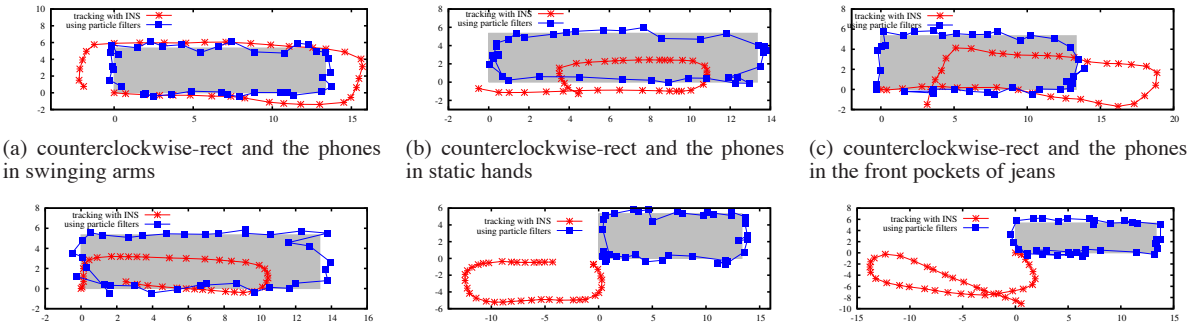
The INS components are always on for the aim of rotating screen automatically, and thus the employment of the INS does not add extra energy consumption. Therefore, XINS is able to save energy consumption.

CONCLUDING REMARKS

We proposed the XINS, which uses particle filters to fuse the signals from the WiFi, indoor maps and inertial navigation system to enable the localization of pedestrian indoor with a mobile device. Our main contributions come in two-folds, namely, the non-intrusive calibration of the accelerometer inside a mobile device, and VEM, used to determine the direction of pedestrian by computing in the dual, energy domain (thanks to the Equipartition theorem), reduces the number of integrals from three to one; and therefore, avoids small errors from being magnified. Besides, we also introduce the landmark detection based on WiFi signal and indoor map to dramatically reduce the number of particles for predicting accurate position. We will continue conducting experiments in more complicated scenarios.

REFERENCES

1. D. Allan and J. Barnes. A modified Allan variance with increased oscillator characterization ability. In *Thirty Fifth Annual Frequency Control Symposium*, pages 470–475. IEEE, 1981.



(a) counterclockwise-rect and the phones in swinging arms

(b) counterclockwise-rect and the phones in static hands

(c) counterclockwise-rect and the phones in the front pockets of jeans

(d) clockwise-rect and the phones in swinging arms

(e) clockwise-rect and the phones in static hands

(f) clockwise-rect and the phones in the front pockets of jeans

Figure 8. The localization results, where the x and y axes are measured in meters. The grey rectangle represents the $5.35m \times 13.35m$ rectangle walking around, red lines show the INS-only tracking results, and blue lines show the locations given by particle filters).

2. M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

3. J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *ICRA 2010*, pages 4379–4384. IEEE, 2010.

4. I. Constandache, R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *INFOCOM 2010*, pages 1–9. IEEE, 2010.

5. I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox. EnLoc: Energy-efficient localization for mobile phones. In *INFOCOM 2009*, pages 2716–2720. IEEE, 2009.

6. M. Cypriani, F. Lassabe, P. Canalda, and F. Spies. Wi-Fi-based indoor positioning: Basic techniques, hybrid algorithms and open software platform. In *IPIN 2010*, pages 1–10. IEEE, 2010.

7. W. Fong, S. Ong, and A. Nee. Methods for in-field user calibration of an inertial measurement unit without external equipment. *Measurement Science and Technology*, 19:085202, 2008.

8. K. Hattori, R. Kimura, N. Nakajima, T. Fujii, Y. Kado, B. Zhang, T. Hazugawa, and K. Takadama. Hybrid indoor location estimation system using image processing and WiFi strength. In *WINSYS 2009*, pages 406–411. IEEE, 2009.

9. D. Kelly, R. Behan, R. Villing, and S. McLoone. Computationally tractable location estimation on WiFi enabled mobile phones. In *ISSC 2009,*, pages 1–6. Iet, 2009.

10. A. Kim and M. Golnaraghi. Initial calibration of an inertial measurement unit using an optical position tracking system. In *PLANS 2004*, pages 96–101. IEEE, 2004.

11. M. Kourogi, T. Ishikawa, and T. Kurata. A method of pedestrian dead reckoning using action recognition. In *PLANS 2010*, pages 85–89. IEEE, 2010.

12. F. Lassabe, P. Canalda, P. Chatonnay, and F. Spies. Indoor Wi-Fi positioning: techniques and systems. *Annals of telecommunications*, 64(9):651–664, 2009.

13. S. Lee and K. Mase. Recognition of walking behaviors for pedestrian navigation. In *CCA 2001*, pages 1152–1155. IEEE, 2001.

14. J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308, 1965.

15. A. Olivares, G. Olivares, J. Gorriz, and J. Ramirez. High-efficiency low-cost accelerometer-aided gyroscope calibration. In *ICTM 2009*, volume 1, pages 354–360. IEEE, 2009.

16. J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *MobiSys'10*, pages 299–314. ACM, 2010.

17. J. Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer Verlag, 2005.

18. O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *UbiComp'08*, pages 114–123. ACM, 2008.

19. Z. Wu, Z. Wang, and Y. Ge. Gravity based online calibration for monolithic triaxial accelerometers' gain and offset drift. In *CICA 2002*, volume 3, pages 2171–2175. IEEE, 2002.

20. Y. Zhang, L. Li, and Y. Zhang. Research and Design of Location Tracking System Used in Underground Mine Based on WiFi Technology. In *IFCSTA 2009*, pages 417–419. IEEE, 2009.

21. Z. Zhuang, K. Kim, and J. Singh. Improving energy efficiency of location sensing on smartphones. In *MobiSys'10*, pages 315–330. ACM, 2010.